Data Sharing by Threads Solutions

Copyright © James Raynard 2021

Data sharing by threads

- What is the main problem that can occur when threads share data?
  - The threads can interfere with each other when modifying the shared data
  - This can cause the shared data to be in an inconsistent or corrupted state

Copyright © James Raynard 2021

Interleaved Execution Example

- Write a program in which the output from multiple threads is interleaved
- Where is the shared data variable in this program?
  - cout is a global object which represents the standard output stream

Copyright © James Raynard 2021

Data race definition

- What is meant by the term "data race" in C++?
  - A data race occurs when multiple threads make potentially conflicting accesses to the same memory location
  - Two or more threads access the memory location and at least one of the threads modifies it
- How can a data race be avoided?
  - The accesses to the shared data must be synchronized in some way
  - Each thread's access is made to "happen before" the next (the shared data is protected with a mutex)
  - Or all accesses are made "atomically" (operations on the shared data cannot be interrupted)

Copyright © James Raynard 2021

Memory Location

- Give some examples of C++ "memory locations"
  - A variable of built-in type
  - A pointer
  - An element in a container
  - A struct or class member which is a scalar object

Copyright © James Raynard 2021

Golden rule of C++ multithreading

- Memorize the golden rule of C++ multithreading
  - If:
  - There is a shared built-in variable, object member, container instance etc
  - And it is accessed by threads which could interleave their execution
  - And at least one of the threads modifies the memory location
  - Then:
  - Use a mutex to protect the shared memory location
  - Or make the shared memory location an "atomic" type

Copyright © James Raynard 2021